

© 2002-2013 Volnys Bernal 1

Cliente UDP

Volnys Borges Bernal
volnys@lsi.usp.br
<http://www.lsi.usp.br/~volnys>



© 2002-2013 Volnys Bernal 2

Agenda

- ❑ Resumo de Chamadas UDP
- ❑ Chamada socket()
- ❑ Chamada connect()
- ❑ Chamada send()
- ❑ Chamada recv()
- ❑ Chamada sendto()
- ❑ Chamada recvfrom()
- ❑ Chamada close()

© 2002-2013 Volnys Bernal 3

Chamadas UDP



© 2002-2013 Volnys Bernal 4

Resumo de Chamadas UDP

Lado Cliente

```

socket ()
connect ()
send ()
recv ()
close ()

```

Lado Servidor

```

socket ()
bind ()
recvfrom ()
sendto ()
close ()

```

Pré define o parceiro de comunicação

© 2002-2013 Volnys Bernal 5

Resumo de Chamadas UDP

Lado Cliente

```

socket ()
sendto ()
recvfrom ()
close ()

```

Lado Servidor

```

socket ()
bind ()
recvfrom ()
sendto ()
close ()

```

Define o parceiro de comunicação a cada chamada

© 2002-2013 Volnys Bernal 6

Chamada socket()



© 2002-2013 Volhys Bernal 7

Chamada socket()

- ❑ **Objetivo**
 - ❖ Criar um novo socket (plug de comunicação)
- ❑ **Resultado**
 - ❖ Retorna um descritor de arquivo.
- ❑ **Sintaxe**

```
sd = socket (int domain, int type, int protocol)
```
- ❑ **Observação:**
 - ❖ Quando um socket é criado, não possui nenhuma informação sobre o parsocket (endereços IPs e portas dos parceiros).
 - ❖ Endereços IP e portas são informados nas chamadas bind() (lado servidor) e connect() (lado cliente).

© 2002-2013 Volhys Bernal 8

Chamada socket()

- ❑ **Sintaxe**

```
#include <sys/socket.h>
int socket(int domain, int type, int protocol)
```

Socket descriptor

Id. do protocolo:

- UDP (17)
- TCP (6)

Pilha de protocolos:

- PF_INET
- PF_INET6
- PF_X25

Tipo da comunicação:

- SOCK_STREAM (TCP)
- SOCK_DGRAM (UDP)
- SOCK_RAW (IP)

© 2002-2013 Volhys Bernal 9

Chamada socket()

- ❑ **Tipo de serviço**
 - ❖ SOCK_STREAM
 - Fluxo de bytes sem delimitação
 - Full duplex
 - Orientada a conexão
 - Comunicação confiável:
 - sem perda de dados, sem duplicação, entrega na ordem
 - Deve ser utilizado com o protocolo TCP
 - Chamadas para transmissão e recepção de dados:
 - read(), write() ou send(), recv()
 - ❖ SOCK_DGRAM
 - Datagrama (mensagens)
 - Deve ser utilizado com o protocolo UDP
 - Chamadas para transmissão e recepção de dados:
 - send(), recv()
 - ❖ SOCK_RAW
 - Permite acesso a protocolos de mais baixo nível
 - Datagrama (mensagens)
 - Chamadas para transmissão e recepção de dados:
 - send(), recv()

© 2002-2013 Volhys Bernal 10

Chamada socket()

- ❑ **Exemplo de criação de socket UDP**

```
#include <sys/socket.h>

int sd; // socket descriptor
...
sd = socket (PF_INET, SOCK_DGRAM, 17);
if (sd == -1)
{
    perror("Erro na chamada socket");
    exit(1);
}
...

```
- ❑ **Obs:**
 - ❖ O valor 17 representa o protocolo UDP, foi obtido consultando /etc/protocols. Pode também ser obtido através de resolução de nomes via getprotobyname().

© 2002-2013 Volhys Bernal 11

Chamada Connect()



© 2002-2013 Volhys Bernal 12

Chamada connect()

- ❑ **Objetivo**
 - ❖ Permite definir uma sessão de comunicação TCP, UDP ou IP
 - ❖ Informa ao sistema o socket address (IP+porta) do parceiro de comunicação
- ❑ **Funcionamento**
 - ❖ Deve ser executado no lado cliente
 - ❖ UDP:
 - Não são enviados datagramas
 - ❖ TCP:
 - Ativa o processo de estabelecimento de conexão TCP

© 2002-2013 Volhys Bernal 13

Chamada connect()

□ Sintaxe

```
#include <netdb.h>
int connect(int sd,
            struct sockaddr *serversockaddr,
            int size)
```

Socket descriptor

Socket address (IP + porta) do servidor

Tamanho da estrutura de endereço (sockaddr_in)

© 2002-2013 Volhys Bernal 14

Chamada connect()

```
#include <netdb.h>

int status; //estado da chamada
struct sockaddr_in serversockaddr; //endereço do servidor
...

// define endereço destino
serversockaddr.sin_family = AF_INET;
serversockaddr.sin_port = htons(serverport);
status = inet_pton(AF_INET, stringIP, &serversockaddr.sin_addr);
if (status <= 0)
    perror("Erro na conversão do endereço IP");

// ativa connect
status = connect( sd,
                (struct sockaddr *)&serversockaddr,
                sizeof(serversockaddr) );

if (status != 0)
    perror("Erro na chamada connect");
```

© 2002-2013 Volhys Bernal 15

Chamada connect

□ Exemplo de iniciação da estrutura sockaddr_in

```
int
char IPstr[15]
```

80

10.0.0.1

htons()

AF_INET

inet_pton()

sockaddr_in

sin_family

sin_port

sin_addr

sin_zero

© 2002-2013 Volhys Bernal 16

Chamada send()



© 2002-2013 Volhys Bernal 17

Chamada send()

□ Função para transmissão de dados

□ Pode ser utilizada por clientes e servidores

```
int send(int sd, void * txbuffer, int msgsize, int flags)
```

Socket Descriptor

Ponteiro para buffer da mensagem (end. do buffer da mensagem)

Tamanho da mensagem

Opções

© 2002-2013 Volhys Bernal 18

Chamada send()

□ Exemplo:

```
char txbuffer[80];

...
status = send( sd, txbuffer, strlen(txbuffer)+1, 0)
if (status < 0)
    perror("Erro na chamada send");
...

```

© 2002-2013 Volhys Bernal 19

Chamada recv()



© 2002-2013 Volhys Bernal 20

Chamada recv()

- ❑ Recebimento de datagramas
- ❑ Pode ser utilizada por clientes e servidores

```
int recv(int sd, void * rxbuffer, int rxbuffersize, int flags)
```

Socket
Descriptor

Tamanho do
buffer

Opções

Ponteiro para o buffer
(end. do buffer de recepção)

© 2002-2013 Volhys Bernal 21

Chamada recv()

- ❑ Exemplo:

```
char rxbuffer[80];
...
status = recv(sd, rxbuffer, sizeof(rxbuffer), 0)
if (status < 0)
    perror("Erro na chamada recv");
printf("MSG recebida: %s\n", rxbuffer);
...
```

© 2002-2013 Volhys Bernal 22

Chamada recv()

- ❑ Bloqueante
 - ❖ Se não existirem mensagens na fila de recepção o processo fica aguardando sua chegada
 - ❖ Exceção: quando o socket for criado como não bloqueante (ver fcntl(2)).
- ❑ Retorno
 - ❖ Se a chamada tiver sucesso, o valor retornado é o tamanho do datagrama

© 2002-2013 Volhys Bernal 23

Chamada close()



© 2002-2013 Volhys Bernal 24

Chamada close()

- ❑ Exemplo:

```
int sd; // socketdescriptor
...
status = close(sd);
if (status == -1)
    perror("Erro na chamada close");
...
```

© 2002-2013 Volhys Bernal 25

Exercício



© 2002-2013 Volhys Bernal 26

Exercício

(1) Identifique a porta utilizada no serviço “echo”.

(2) Implemente um cliente para o serviço “echo” utilizando o protocolo UDP.

- ❖ O serviço echo responde exatamente com a seqüência ASCII recebida.

© 2002-2013 Volhys Bernal 27

Exercício

(3) Identifique a porta utilizada no serviço “daytime”.

(4) Implemente um cliente para o serviço daytime utilizando o protocolo UDP.

- ❖ O serviço daytime UDP responde com a data e hora do servidor no instante de recebimento do datagrama UDP.

© 2002-2013 Volhys Bernal 28

Referências Bibliográficas



© 2002-2013 Volhys Bernal 29

Referências Bibliográficas

- **COMMER, DOUGLAS; STEVENS, DAVID**
 - ❖ Internetworking with TCP/IP: volume 3: client-server programming and applications
 - ❖ Prentice Hall
 - ❖ 1993